An improved version of the augmented epsilon-constraint method (AUGMECON2) for finding the exact Pareto set in Multi-Objective Integer Programming problems

Mavrotas, George and Florios, Kostas

National Technical University of Athens

2013

An improved version of the augmented ε-constraint method (AUGMECON2) for finding the exact Pareto set in Multi-Objective Integer Programming problems

George Mavrotas, Kostas Florios

Laboratory of Industrial and Energy Economics, School of Chemical Engineering,
National Technical University of Athens, Zographou Campus, Athens 15780, Greece, Tel: +30 210-7723202,
fax: +30 210 7723155, e-mail: mavrotas@chemeng.ntua.gr

Abstract: Generation (or a posteriori) methods in Multi-Objective Mathematical Programming (MOMP) is the most computationally demanding category among the MOMP approaches. Due to the dramatic increase in computational speed and the improvement of Mathematical Programming algorithms the generation methods become all the more attractive among today's decision makers. In the current paper we present the generation method AUGMECON2 which is an improvement of our development, AUGMECON. Although AUGMECON2 is a general purpose method, we will demonstrate that AUGMECON2 is especially suitable for Multi-Objective Integer Programming (MOIP) problems. Specifically, AUGMECON2 is capable of producing the exact Pareto set in MOIP problems by appropriately tuning its running parameters. In this context, we compare the previous and the new version in a series of new and old benchmarks found in the literature. We also compare AUGMECON2's performance in the generation of the exact Pareto sets with established methods and algorithms based on specific MOIP problems (knapsack, set packing) and on published results. Except from other Mathematical Programming methods, AUGMECON2 is found to be competitive also with Multi-Objective Meta-Heuristics (MOMH) in producing adequate approximations of the Pareto set in Multi-Objective Combinatorial Optimization (MOCO) problems.

Keywords: Multi-Objective Programming, ε-constraint method, exact Pareto set

1. Introduction

The rapid improvement in computer performance, software and algorithms transformed the almost unsolvable calculation problems of previous decades into trivial tasks. This is especially true for Mathematical Programming where problems with thousand of variables and constraints can be solved in seconds or minutes. It is well known that the Multiple Objective Mathematical Programming refers to the solution of Mathematical Programming problems with more than one objective functions. Given that usually there is no unique optimal solution (optimizing simultaneously all the objective functions), the aim is to find the most preferred among the Pareto optimal solutions [1]. Therefore, MOMP methods have to combine optimization with decision support.

In the late seventies Multiple Objective Mathematical Programming methods were classified by Hwang and Masud into three classes according to the phase in which the decision maker was involved in the decision making process [2]: The a priori methods, the interactive methods and the a posteriori or generation methods. The latter class was somehow neglected at this time as it was the most computationally demanding and only small problems could be addressed mostly with academic software.

Nowadays, with the vast improvement in computer power the generation approaches become all the more popular as they have some significant advantages. The solution process can be divided into two separate phases: Phase one is the generation of the Pareto optimal solutions (all or a subset of them). Phase two is the subsequent involvement of the decision maker when all the information is on the table. No re-optimizations and further interaction are needed (especially important whenever the DM is hardly available) and the fact that none of the potential solutions has been left undiscovered, reinforces the DM's confidence on the final decision.

AUGMECON is a generation method introduced by Mavrotas in [3]. It is an improvement of the original ε-constraint method which is along with the weighting method one of the two most popular methods for generating representations of the Pareto front. As it is described in [3], the ε-constraint method has certain advantages in relation to the weighting method especially in the presence of discrete variables (Mixed Integer or Pure Integer problems). In the current work we are going one step further, introducing AUGMECON2 an improvement of AUGMECON that exploits the information from the slack variables in every iteration. The improvements regard the reduction in computation time as many redundant iterations are avoided.

These improvements are more effective when the problem contains discrete variables and the feasible region is non-convex. AUGMECON2 proved to be very efficient in Multi-Objective Integer Programming (MOIP) problems where the Pareto set is finite and countable. In this kind of problems we can adjust the method in order to produce all the Pareto optimal solutions. In the literature, several versions of the ε-constraint method have been appeared trying to improve its performance or adapt it to a specific type of problems like MOIP problems (see e.g. [4-6]).

This is extremely important as the optimization community is interested in methods producing the exact Pareto set in Multi-Objective Combinatorial Optimization (MOCO) problems (see e.g. [7-9]). Various methods have been proposed, either generic or specific (for specific MOCO problems like e.g. the knapsack problem) that are able to find all the Pareto optimal solutions in MOCO problems. This kind of problems can be formulated as mathematical programming problems and more specifically as MOIP problems (usually with only 0-1 variables).

In the last decade a rapid growth on the Multi-Objective Meta-Heuristic (MOMH) methods has been observed. These methods provide an approximation of the Pareto front using multi-objective versions of established metaheuristics (e.g. genetic algorithms, tabu search, simulated annealing among others). The MOMH algorithms are usually compared in terms of speed and coverage. The latter can be briefly defined as the portion of the true (exact) Pareto front that they are able to discover [8]. Therefore there is a need for calculating the exact Pareto front in big MOCO problems in order to use them as benchmarks for MOMH algorithms. This makes the present work for calculating the exact Pareto set in MOIP and MOCO problems even more important.

The rest of the paper is organized as follows: Section 2 reviews the related literature. In Section 3 the novel parts of the AUGMECON2 method are described and the application of the proposed method in the generation of the exact Pareto front in Multiobjective Integer Programming problems is illustrated. In Section 4 the computational experiment for the evaluation of the proposed method in MOIP test problems is described. The results of the comparison of AUGMECON2 with other methods (including its older version) in a variety of new and existing in the literature test problems are discussed in Section 5. In Section 6 we apply the method in an illustrative example of project selection. Finally in Section 7 the basic concluding remarks are discussed.

2. Related literature

A non exhaustive literature review for the general MOIP problem, as well as for specific MOCO problems that we will use in our work, namely the Multi-Objective version of the Multidimensional Knapsack Problem (MKP) and the Bi-Objective Set Packing Problem (BOSPP) is presented.

The general Multi-Objective Integer Programming (MOIP) problem can be formulated as in Chinchuluun and Pardalos [10]

$$\begin{aligned} &\max \ Cx \\ &s.t. \quad Ax \le b \\ &\qquad x \ge 0 \qquad x \in \mathbb{Z}^n \end{aligned} \tag{1}$$

where C is a $k \times n$ matrix, A is a $m \times n$ matrix and b is a m vector

In the following, the discussion regarding MOIP solution techniques is focused specifically on the class of generation methods. Early work on MOIP starts with Bitran in late '70s [11, 12] who developed algorithms for solving multiobjective linear programs with binary variables based on enumerative schemes. Klein and Hannan [13] proposed a sequential method to generate all efficient points of the multiobjective integer programming problem. Their method included the solution of a series of integer linear programs where new constraints which depend on previously discovered efficient points are added for all but one objectives. Chalmet et al. [14] studied the MOIP problem for the biobjective case using the well known weighted sum method. Since the weighted sum method provides only the supported efficient solutions, the authors include an additional constraint which ensures access also to the unsupported efficient solutions. Jahanshahloo et al. [15] proposed a method to find all efficient solutions of 0-1 MOLP problem. These solutions are specified without generating all feasible solutions. In every iteration of the algorithm, for obtaining some efficient solutions of 0-1 MOLP, a 0-1 single objective LP problem is solved. Also, Sourd and Spanjaard [16] developed a multiobjective Branch-and-Bound framework for MOCO problems. The bounding in their method is performed via a set of points rather than a single ideal point. Numerical experiments regard the bi-objective spanning tree problem and the method handles easily problems up to 500 nodes. Ralphs et al. [17] implemented an improved algorithm based on the weighted Tchebycheff scalarization and provided evidence on the biobjective knapsack problem and a capacitated network routing problem. The open source SYMPHONY MIP solver is utilized in their approach. An interesting approach is that of Ozlen and Azizoglu [18], who presented a general method to generate all non-dominated solutions of a MOIP problem. Their approach is based on the identification of objective efficiency ranges, which are identified by solving simpler MOIP problems with fewer objectives. Their method is presented for a tri-objective integer programming problem and is generalized to a MOIP problem with k objectives. Also, Przybylski et al. [19] presented a generalization of the two phase method to solve MOIP with more than two objectives. It is well known that phase one aims at computing all supported efficient solutions. The authors generalized the second phase which aims at computing the non-supported efficient points, a task which is difficult to describe in the general multi-objective case. Experimental results of the method are given for the three-objective assignment problem.

A stream of research on MOIP utilizes the ε-constraint method [20] with some enhancements for the generation of the efficient solutions. Laumanns et al. [5] implemented an adaptive scheme for varying the right hand side of the ε-constraint method, and called their method adaptive ε-constraint. The complexity of their method is $O(k^{m-1})$ where k is the number of Pareto optimal solutions and m is the number of objectives. Laumanns et al. [5] tested their approach on the multiobjective

multidimensional knapsack problem. Also, the adaptive ε-constraint method has been tested in Rayside et al. [21]. Ehrgott and Ruzika [22] proposed an improved ε-constraint method. They criticized the traditional ε-constraint method for two weaknesses: the lack of easy-to check conditions for properly efficient solutions and the inflexibility of the constraints. Then, they presented two modifications which addressed these weaknesses. First they included slack variables in the formulation and second they elasticized the constraints by including surplus variables. The improved ε-constraint method they proposed combined both modifications. Nevertheless, the authors did not give any computational results in their study. Finally, Mavrotas [3] presented an effective implementation of the ε-constraint method for generating the Pareto optimal solutions in a multiobjective mathematical program. The author proposed a novel version of the method, augmented ε-constraint method (AUGMECON) which avoids the generation of weakly Pareto optimal solutions and accelerated the whole process by avoiding redundant iterations. The method AUGMECON has been implemented in GAMS, a widely used modeling language, and has already been used in several applications.

Other scalarization techniques for MOIP include the methods of Sylva and Crema [23], [24], Lokman [25] and Koksalan and Lokman [26]. Sylva and Crema in [23] presented an algorithm for enumerating all non-dominated solutions of multi-objective integer programming problems. Employing a straightforward theoretical approach, the problem is solved using a sequence of progressively more constrained integer programs, thus generating a new solution at every step. This algorithm is suitable for medium sized problems, because at every step new constraints and binary variables are added for every computed efficient solution. Subsequently, Sylva and Crema in [24] presented an algorithm for generating a subset of non-dominated vectors of multi-objective mixed integer programming problems. Starting from an initial non-dominated vector, the procedure finds at every iteration a new vector that maximizes the infinity-norm distance from the set dominated by the previously found solutions. This algorithm works also for MOLP as well as MOIP problems. Lokman in her MSc thesis [25] proposed two algorithms for the exact solution of MOIP. The first exact algorithm is an improvement of the algorithm of Sylva and Crema [23], since the number of binary variables and constraints introduced to the model for each new efficient solution is decreased. The second exact algorithm is quite different and employs searching and sorting to generate the efficient solutions. In the computational results, Lokman presents evidence from the multiobjective multidimensional knapsack problem that Algorithm 2 is much more efficient than Algorithm 1, while Algorithm 1 is already much more efficient than Sylva and Crema [23] exact method. Additionally, specifically for Algorithm 2, results from multiobjective minimum spanning tree and multiobjective shortest path problems are presented. The largest instance of multiobjective multidimensional knapsack solved is 3 objectives, 3 constraints and 200 items, solved in 184,608.70 sec with 27,260 efficient solutions, solved by Algorithm 2 (e.g. see [25], pp. 56). Koksalan and Lokman in [26] present a heuristic approach to approximate the nondominated frontier of MOIP by fitting smooth hypersurfaces. For a given problem, they fit the hypersurface using a single non-dominated reference vector. The authors experimented with different types of MOIP problems and demonstrated that in all cases the fitted hypersurfaces approximate all nondominated vectors well. They discuss that such an approximation is useful to find the neighborhood of preferred regions of the nondominated vectors with very little computational effort. Further computational effort can be spent in the identified region to find the actual nondominated vectors the decision maker will prefer. The interested reader is referred to excellent review papers on MOIP (including interactive methods and specific combinatorial problems), e.g. Teghem and Kunsch [27], Ulungu and Teghem [28], Rasmussen [29], Ehrgott and Gandibleux [30] and Alves and Climaco [31].

The specific MOIP problems tackled in this paper are the Multi-Objective Multidimensional Knapsack Problem and the Bi-objective Set Packing Problem, briefly discussed in the following.

The Multi-Objective Multidimensional Knapsack Problem (MOMKP) can be formulated as in [9] and [32-38].

$$\begin{aligned} &\max \ Cx \\ &s.t. \quad Ax \le b \\ &\qquad x \ge 0 \qquad x \in \{0,1\}^n \end{aligned} \tag{2}$$

where C is a $k \times n$ matrix, A is a $m \times n$ matrix and b is a m vector

Zitzler and Thiele in [33] first introduced the MOMKP and proposed a strength pareto evolutionary algorithm (SPEA) to solve it. They also introduced the well known instances with k=m=2,3,4 and n=100,250,500,750 (12 instances). The elements of C and A are uncorrelated random numbers from Uniform[10,100] and b corresponds to a tightness ratio of 50%. The instances of this paper have been solved by numerous papers and are used also in our computational study. Instances with larger coefficients, thus from Uniform[100,1000], and also correlation between matrices C and A have been created by us for further experiments and they are available along with their solutions (exact Pareto fronts) in the web site: https://sites.google.com/site/kflorios/augmecon2

Florios et al. in [9] solved exactly instances of the MOMKP with k=m=3 and n=10 to 50 using the Multicriteria Branch and Bound (MCBB) method and suitable branching and construction heuristics. The authors also solved approximately the same instances using SPEA2 and NSGAII. Comparison included the adaptive ε-constraint (ADECON) method of Laumanns et al. [5]. MCBB with suitable heuristics was found to be noticeably faster than adecon in small and medium instances (up to 50 items). Nevertheless, as the size of the problem increased, the advantage of MCBB over adecon deteriorated. In the present paper, a larger instance of k=m=3 and n=100 (not solvable with MCBB plus heuristics) is solved with AUGMECON in a fraction ($^1/_8$) of the run time needed by ADECON in the past (see Laumanns et al. [39]). Recently, Shah and Reed in [36] proposed the epsilon-nondominated hierarchical Bayesian optimization algorithm (ε-hBOA) for the MOMKP and compared to SPEA2 and ε-NSGAII. The authors also introduced for the first time weakly correlated coefficients for A and C. They did not account though for larger coefficients, drawing A and C coefficients typically from Uniform[10,100] as in Zitzler and Thiele in [33]. Finally, Lust and Teghem in [37] present an analytical survey of the MOMKP along with its solution methods (especially metaheuristics) and propose a new approach, the two-phase Pareto local search (2PPLS) adapted to the MOMKP. The authors provide computational results in the Zitzler and Thiele data found in [33]. Nevertheless, the simulations do not account for larger coefficients and/or weakly correlated data.

Regarding the second MOIP problem treated in this paper, it is well known that the single objective Set Packing Problem (SPP) is a classic combinatorial optimization problem which is among the first ones to be proven NP-complete [40]. The Bi-objective Set Packing Problem (BOSPP) has been introduced by Delorme et al. [41]. Also, Tricoire [42] has applied his Multi-Directional Local Search method to the BOSPP. There are benchmarks for this problem by Delorme et al. [41], available at the MOCOlib website maintained by X.Gandibleux. The BOSPP is less studied in the literature compared with the MOMKP.

3. Methodological part

3.1 The improved version of the augmented ε-constraint (AUGMECON2)

We start this section by briefly describing the original version of the augmented ε-constraint method, named AUGMECON [3] for the integrality of the paper. AUGMECON enhances the conventional ε-constraint method for generating the Pareto optimal solutions in Multi-Objective Mathematical Programming problems. It is well known that the ε-constraint has certain advantages in relation to the weighting method as described in [3]. AUGMECON addresses some weak points of the conventional ε-constraint, namely, the guarantee of Pareto optimality of the obtained solution in the payoff table as well as in the generation process and the increased solution time for problems with several (more than two) objective functions.

In the original AUGMECON method the problem solved is the following:

$$
\begin{aligned}
&\max\ (f_1(\mathbf{x}) + eps\times (S_2/r_2 + S_3/r_3 + \ldots + S_p/r_p)) \\
&\text{st} \\
&f_2(\mathbf{x}) - S_2 = e_2 \\
&f_3(\mathbf{x}) - S_3 = e_3 \\
&\ldots \\
&f_p(\mathbf{x}) - S_p = e_p \\
&\mathbf{x} \in S \ \text{ and } \ S_i \in R^+
\end{aligned}
\tag{3}
$$

where e_2, e_3,...,e_p are the parameters for the RHS for the specific iteration drawn from the grid points of the objective functions 2,3,...,p. The parameters r_2, r_3,...,r_p are the ranges of the respective objective functions. S_2, S_3,..., S_p are the surplus variables of the respective constraints and $eps \in [10^{-6}, 10^{-3}]$.

In AUGMECON2, the improved version of AUGMECON we slightly modify the objective function as follows:

$$\max\ (f_1(\mathbf{x}) + eps\times (S_2/r_2 + 10^{-1}\times S_3/r_3 + \ldots + 10^{-(p-2)}\times S_p/r_p))$$

This modification is done in order to perform a kind of lexigographic optimization on the rest of the objective functions if there are any alternative optima. For example, with this formulation the solver will find the optimal for f_1 and then it will try to optimize f_2, then f_3 and so on. With the previous formulation the sequence of optimizations of f_2 to f_p was indifferent, while now we force the sequential optimization of the constrained objective functions (in case of alternative optima).

As it is explained in [3], for each objective function *2...p* we calculate the objective function range. Then we divide the range of the k-th objective function to q_k equal intervals using $(q_k\text{-}1)$ intermediate equidistant grid points. Thus we have in total (q_k+1) grid points that are used to vary parametrically the RHS (e_k) of the k-th objective function. The total number of runs becomes $(q_2+1) \times (q_3+1) \times \ldots \times (q_p+1)$. Let r_k be the range of the objective function k (k=2...p). Then the discretization step for this objective function is given as:

$$step_k = r_k/q_k$$

The RHS of the corresponding constraint in the t-th iteration in the specific objective function will be:

$$e_{kt} = fmin_k + t \times step_k$$

where $fmin_k$ is the minimum from the payoff table and t the counter for the specific objective function.

In each iteration we check the surplus variable that corresponds to the innermost objective function. In this case it is the objective function with p=2. Then we calculate the *bypass coefficient* as:

$$b = int(S_2/step_2)$$

where *int( )* is the function that returns the integer part of a real number. When the surplus variable S_2 is larger than the $step_2$, it is implied that in the next iteration the same solution will be obtained with the only difference being the surplus variable which will have the value S_2-$step_2$. This makes the iteration redundant and therefore we can bypass it as no new Pareto optimal solution is generated. The bypass coefficient b actually indicates how many consecutive iterations we can bypass.

This can be shown with the following example: Assume that we have a three-objective problem with the following payoff table (all objective functions to be maximized):

Table 1. Payoff table

	$f_1(x)$	$f_2(x)$	$f_3(x)$
max $f_1(x)$	980	796	803
max $f_2(x)$	836	876	765
max $f_3(x)$	809	821	905

From the payoff table we have r_2=80 and r_3= 140. We then divide the two ranges in 10 equal intervals with $step_2$=8 και $step_3$=14. The AUGMECON2 process is the following:

For i=0 to 10
  e_3=765 + i×14
  For j=0 to 10
    e_2=796 + j×8
    Solve (3)
  Next j
Next i

The objective function $f_2(x)$ is the innermost loop (j counter). Assume that we are in the 2nd outermost (where i=1) and the 5th innermost iteration (where j=4) where e_3=779 and e_2=828 which are the shaded cells in Table 2.

Table 2. Grid points of the problem

obj. function	counter	grid points										
		0	1	2	3	4	5	6	7	8	9	10
$f_2(x)$	j	796	804	812	820	828	~~836~~	~~844~~	852	860	868	876
$f_3(x)$	i	765	779	793	807	821	835	849	863	877	891	905

After the optimization we obtain S_2=18 and S_3=9, which mean that in this iteration the value for the second objective function is:

$f_2 = e_2 + S_2$ = 828+18 = 846 and $f_3 = e_3 + S_3$ = 779+9 = 788.

We conclude that it is redundant to perform the next two iterations with j=5 and j=6 (strikethrough in table 2) because we will result in the same Pareto optimal solution with f_2=846. The only difference is that the surplus variables will be 10 (=18-8) and 2 (18-2×8) for j=5 and j=6 respectively. Therefore we

can bypass these two iterations and go directly from j=4 to j=7 (value for e_2=852 in Table 2). The bypass coefficient b is calculated as $b = int(18/8) = 2$.

The flowchart of the new algorithm is shown in Figure 1:

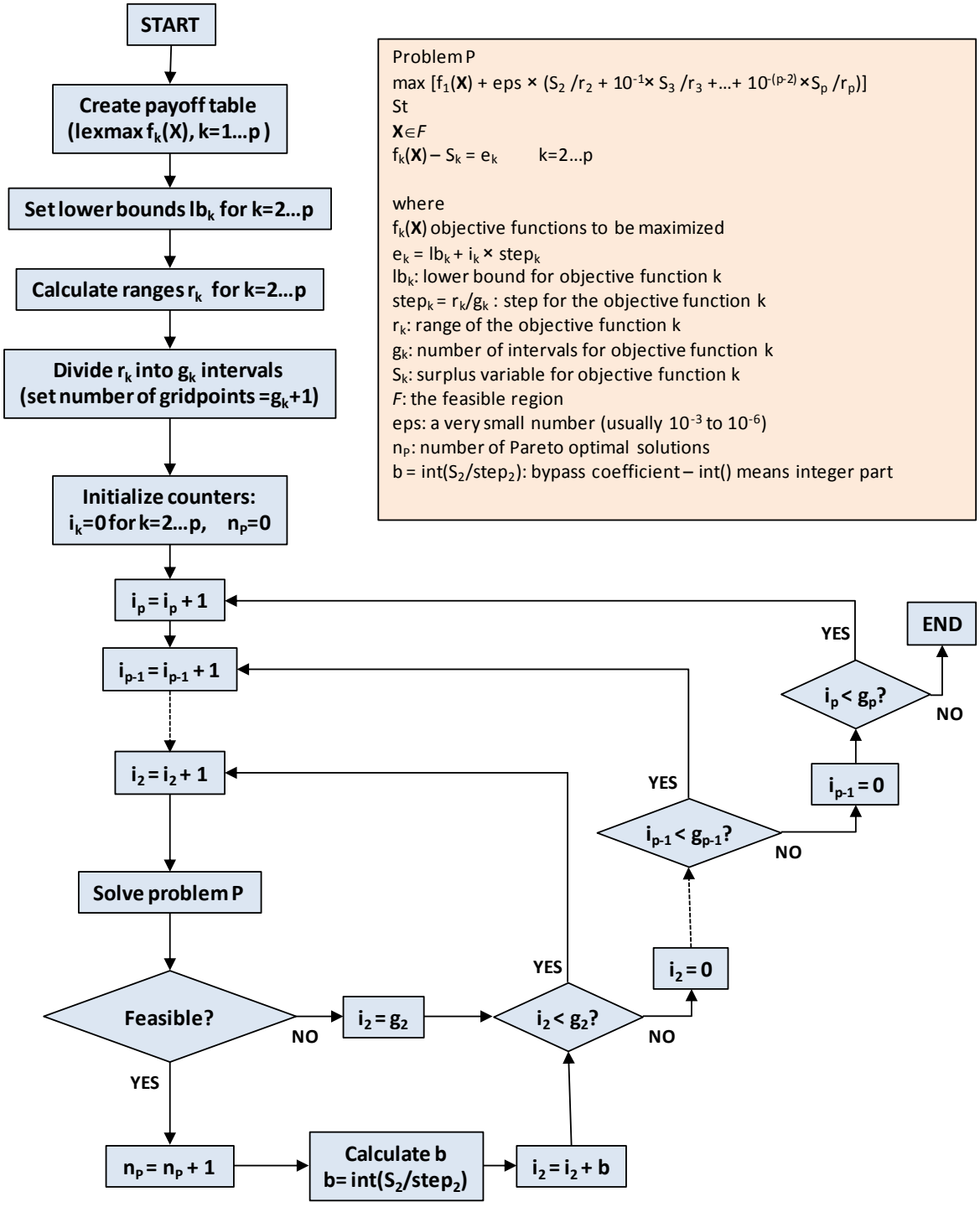

Figure 1. *Flowchart of the AUGMECON2 method*

In this way, using the bypass coefficient to exploit the information from the slack/surplus variables of the constrained objective functions we greatly accelerate the algorithm as we avoid redundant iterations. As we will see in the subsequent section the "jumps" caused in the innermost loop become

even more significant when we reduce the step size (increase the grid density). Therefore, we take advantage of this property in order to produce the exact Pareto front in MOIP problems in reasonable computation time. The GAMS code with some instructions and instances/results of this paper are available in https://sites.google.com/site/kflorios/, while a representative model is available in GAMS model library http://www.gams.com/modlib/libhtml/epscmmip.htm .

3.2 Calculation of the exact Pareto set in MOIP problems using AUGMECON2

As it was mentioned, the ε-constraint method can be properly used for the generation of the Pareto optimal solutions in MOMP problems with discrete variables. In the case of MOIP and 0-1MOMP problems (MOIP problems with only 0-1 variables as integer variables which are the vast majority of MOIP problems) the ε-constraint method can be used to produce the exact (or complete) Pareto set, which means all the Pareto optimal solutions. As it is well known the size of the Pareto set is finite in MOIP problems and the AUGMECON2 is therefore suitable for generating the exact Pareto set for these problems. The only conditions are the following:

1. The objective function coefficients must be integer
2. The nadir points of the Pareto set must be known

In order to relax the first condition we can easily transform the problem to have integer objective function coefficients by multiplying with the appropriate power of 10. For example if we have one decimal digit in the objective function coefficients, we multiply them with 10, if we have two decimal digits we multiply them with 100. It must be noted that the greater the magnitude of the objective function coefficients the more time consuming is the generation of the exact Pareto set (see e.g. section 4 in the computational experiment). The reason is that the range of the objective functions is inevitably greater.

In order to relax the second condition it is adequate to know a lower bound of the nadir point (for maximization problems). The payoff table in multi-objective programming provides the individual optima of the objective functions (in the diagonal). However, it is well known (see e.g. Isermann and Steuer [43]) that it does not guarantee the calculation of the nadir points except only for problems with just two objective functions. For problems with three and more objective functions the minimum of the payoff table's columns do not necessarily provide the nadir point of the problem. Consequently, in order to be sure that we have the true nadir points we cannot rely on the values of the payoff table. We must either calculate them using one of the methods in the literature, or estimate some lower bounds of them. The latter usually results in slightly higher computational effort afterwards, during the AUGMECON2 process. The closer is the lower bound to the nadir point the less is the computational time as the objective function range becomes narrower.

In order to calculate the nadir point several approaches have been proposed in the literature (see e.g. [43-47]). In the case of MOIP problems the procedure of Jorge [47] is suitable to our case. Specifically, one can just reverse the direction of optimization of the *p-1* objective functions and optimize them over the integer efficient set, obtaining thus the elements of the nadir vector. The calculation of the nadir vector can be considered as a pre-processing phase to our method that can be triggered whenever needed.

The computational strategy for calculating the exact Pareto set in MOIP problems with integer objective function coefficients is the following (without loss of generality, assume that all the objective functions are to be maximized):

1. Calculate the objective function ranges of the *p-1* objective functions. This means that we have to calculate or estimate lower bounds for the nadir points. If the nadir points are not given (e.g. in the literature for the specific problem), we may apply an appropriate method (43-47].
2. Assume that the objective function range for the *k*-th objective function is r_k (integer). We select for each objective function a unity step so that for each one of them the number of grid points is exactly r_k+1
3. We apply AUGMECON2 and obtain the exact Pareto set. The unity step size and the calculation (or approximation with lower bounds) of the nadir point guarantee that no Pareto optimal solutions are left undiscovered.

The above calculation procedure is straightforward and can be easily coded either using an external solver with a programming language or in a modeling language. In our case it is coded in GAMS modeling language [48].

4. Computational experiment

4.1 Two objective MKP

The improved augmented epsilon constraint with jumps (AUGMECON2) presented in this study is compared to the original version of augmented epsilon constraint (AUGMECON) developed in [3], firstly in a test bed of 16 artificial datasets (in total, 32 runs). The structure of these datasets is illustrated in Table 3.

Table 3. The test bed of 16 datasets for two objective MKP

U instances		W instances	
2 digits for C	3 digits for C	2 digits for C	3 digits for C
2kp100	2kp100b	2kp-W-100	2kp-W-100b
2kp250	2kp250b	2kp-W-250	2kp-W-250b
2kp500	2kp500b	2kp-W-500	2kp-W-500b
2kp750	2kp750b	2kp-W-750	2kp-W-750b

There are 3 parameters in the creation of Table 3 datasets: a) Type of instances b) digits for C matrix and c) number of items, n

a) Type of instances. In the literature only the U type of instances is present ("U" stands for Uncorrelated instances for C and A matrix elements) and especially with 2 digits for C matrix. In this type belong the mostly used data of Zitzler and Laumanns available at [33] and [49] (which appear in the web site http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/) that have been extensively used in the literature. These are called 2kp100, 2kp250, 2kp500 and 2kp750 in our study. They comprise two objectives, two constraints and n=100,250,500,750. These are the standard benchmarks. U type refers to C matrix elements derived from a Uniform distribution, either in U[10,100] (2 digits) or U[100,1000] (3 digits).

In addition to U type instances, we have added according to Shah and Reed in [36] the new type of "W" instances, which stands for Weakly correlated instances for C and A matrix elements. Following Shah and Reed in [36] we have also introduced correlations between C and A matrix elements. For

each knapsack i, the technological coefficients a_{ij} are generated randomly between 10 and 100 (e.g. $a_{ij} \in U[10,100]$) and the objective function coefficients $c_{ij} \in U[a_{ij} - 10, a_{ij} + 10]$. This correlation level is a realistic factor because e.g. in project selection settings it is expected that probably higher cost candidate projects may have higher evaluation score in some quality criteria. In general the W type problems are harder to solve than the U type. These datasets with their exact Pareto fronts are available in https://sites.google.com/site/kflorios/augmecon2.

b) Digits for C matrix. It is expected that when larger coefficients for the objective functions are present, the MOIP problem becomes more difficult to be solved exactly, since the pay off table results in an order of magnitude greater range between ideal and nadir point co-ordinates. So, also instances with 3 digits for C matrix elements have been studied which come from the corresponding 2 digit instances by simply adding a small error term $e \in U[0,9]$ and multiplying the scale of the 2 digit instances by a factor of 10. So, the structure of each instance is not altered by a different sampling. We note that only the C coefficients have been refined to 3 digits precision. The A coefficients remain in 2 digits precision, so the constraints and thus feasible region of MOIP remain unchanged. We have only increased precision in the measurement of the objective functions. This makes the 3 digits instances harder to solve exactly. This is also true even for Dynamic Programming algorithms in single objective single constraint knapsack problems as stated in an example in Papadimitriou and Steiglitz [50], pp.424-425.

c) Number of items, n. Following Zitzler and Laumanns standard datasets, we have restricted our study in the range n=100, 250, 500, 750 items. Also, larger samples have been created with 1000, 1250 and 1500 items, but were not solved because the exact solution of the 750 item benchmarks was already time consuming, and we chose to expand to factors a) and b) previously mentioned and not just expand the size of the instance, n, keeping only U type and 2 digits instances into account. For U type and 2 digits for C coefficients instances, (the Zitzler-Laumanns benchmarks) Lust has made available also the datasets and the Pareto Fronts at his site for the two objectives case https://sites.google.com/site/thibautlust/research/multiobjective-knapsack

4.2 Three objective MKP

The three objective MKP was analysed using only 3 datasets which are illustrated in Table 4.

Table 4. The three datasets for three objective MKP

	2 digits for C
U instances	3kp40
	3kp50
	3kp100

The dataset coding 3kpX means knapsack problems with three objective functions, three constraints and X binary variables. The instances 3kp40 and 3kp50 are taken from Laumanns et al. [5] while 3kp100 is taken from Laumanns et al. [39] and also the Zitzler-Laumanns TestProblemSuite site mentioned in Section 4.1. The instances 3kp40 and 3kp50 have been also solved in Florios et al. [9]. The main interest here is the solution of 3kp100 which is rather challenging. According to Laumanns et al. in [39], 255h (=hours) was the run time for this instance with adaptive epsilon constraint. In this study, the AUGMECON2 method proposed improves greatly in this run time, being roughly an order of magnitude faster.

4.3 Two objective SPP

The formulation of the Bi-objective Set Packing Problem (BOSPP) is as follows:

$$\begin{aligned} &\max \sum_{j=1}^{n} c_j^{(1)} x_j \\ &\max \sum_{j=1}^{n} c_j^{(2)} x_j \quad st \\ &\sum_{j=1}^{n} t_{ij} x_j \le 1 \quad i = 1,...,m \\ &x_j \in \{0,1\} \quad j = 1,...,n \end{aligned} \tag{4}$$

Number of constraints is m, number of variables is n, the objective function 1 coefficients are $c^{(1)}$, the objective function 2 coefficients are $c^{(2)}$ and the matrix t_{ij} is as follows. For each constraint i=1,…,m in matrix t, it is t_{ij}=1 if variable j=1,…,n is involved in the i=1,…,m constraint. Else, t_{ij}=0. Matrix t is a sparse matrix, containing 0-1 elements t_{ij}. The max number of 1s in a row i of matrix t is the parameter max-one. Also, note that the RHS of the ≤ constraints in model (4) is 1 (set packing constraints) and this is a vector maximization problem (BOSPP).

AUGMECON2 presented in this study is compared to the dichotomic procedure with additional constraints ([41], [51] and [52]), in a collection of 120 benchmarks for the BOSPP available at the site http://www.emse.fr/~delorme/SetPacking.html#BOSPP. These datasets are also available via the MOCO lib web site http://xgandibleux.free.fr/MOCOlib/. The structure of these datasets is illustrated in Table 5. For every Number ranging from 1 to 20, there exist 6 types of instances, namely A,B,C,D,E,F so there are totally 20×6=120 instances.

Table 5. The 120 benchmarks for the BOSPP [41]

No	Instances	# Variables	# Constraints	Density (%)	Max-One
1	*2mis100_300*	100	300	2.00	2
2	*2mis100_500*	100	500	2.00	2
3	*2mis101_300*	100	300	2.00	2
4	*2mis101_500*	100	500	2.00	2
5	*2mis200_1000*	200	1000	1.00	2
6	*2mis200_600*	200	600	1.00	2
7	*2mis201_1000*	200	1000	1.00	2
8	*2mis201_600*	200	600	1.00	*2*
9	*2spp100_300*	100	300	3.08	4
10	*2spp100_500*	100	500	2.96	4
11	*2spp101_300*	100	300	2.97	4
12	*2spp101_500*	100	500	3.03	4
13	*2spp200_1000*	200	1000	1.49	4
14	*2spp200_600*	200	600	1.50	4
15	*2spp201_1000*	200	1000	1.49	4
16	*2spp201_600*	200	600	1.49	4
17	*2spp202_1000*	200	1000	2.48	8
18	*2spp202_600*	200	600	2.48	8
19	*2spp203_1000*	200	1000	2.49	8
20	*2spp203_600*	200	600	2.56	8

5. Results and Discussion

The MOMKP model and the augmecon2 method proposed in this paper have been created and solved in GAMS 23.5 environment using CPLEX 12.2 solver. The OS is Windows 7 32-bit and the hardware is a standard core i3 notebook at 2.13 GHz with 4GB RAM for the 5.1 Subsection runs (two objectives MKP) and a standard core i5 notebook at 2.40GHz with 4GB RAM for the 5.2 Subsection runs (three objectives MKP), the 5.3 Subsection runs (BOSPP) and the 5.4 Subsection runs (approximate solution of MOMKP).

5.1 Two objective MKP results

The results from the two objective multidimensional knapsack problems are shown in Table 6. The Grid points are actually the second objective function's range increased by one. The Models solved are the number of IP problems solved for each problem, i.e. the model described in (3). |PF*| is the cardinality of the Pareto set (number of Pareto optimal solutions in the exact Pareto Front)

Table 6. AUGMECON2 and AUGMECON statistics for 2-objective MOMKP

Problem type	Problem dimensions	Grid points	|PF*|	Models solved		CPU time sec	
				AUGMECON	AUGMECON2	AUGMECON	AUGMECON2
U Type [10,100]	2kp100	823	121	823	144	227	40
	2kp250	2534	568	2534	594	2781	932
	2kp500	4176	1416	4176	1654	15290	9601
	2kp750	7232	3030	7232	3699	55483	43355
U Type [100,1000]	2kp100b	8225	135	8225	136	2457	48
	2kp250b	25341	732	25341	746	30621	1914
	2kp500b	41774	2332	41774	2396	145686	17132
	2kp750b	71966	5868	-	6143	-	92913
W Type [10,100]	2kp-W-100	278	111	278	111	568	309
	2kp-W-250	666	375	666	375	4593	4251
	2kp-W5-00	1944	834	1944	968	9052*	7514*
	2kp-W-750	1914	1251	1914	1556	13562*	12579*
W Type [100,1000]	2kp-W-100b	2803	197	2803	205	6765	816
	2kp-W-250b	7267	967	7267	1678	59831	24032
	2kp-W-500b	19527	2980	19527	3256	130022*	46569*
	2kp-W-750b	19619	4960	-	5529	-	128454*

The asterisk (*) indicates that four threads of CPLEX have been used for the IP sub-problems (otherwise only one thread of CPLEX has been used). Dashes "-" in the cells mean that the specific runs did not terminate after 48 hours.

Table 6 reports on the CPU time, Grid points, Models Solved and |PF*| of the corresponding datasets. By looking at the first part of Table 6 results, we see a significant decrease of CPU time and Models Solved in favour of AUGMECON2 over original AUGMECON. This advantage is attributed to the jumps introduced (see section 3.1) which skip redundant gridpoints. This is even more apparent in the second part of Table 6, where the digits of C matrix are three (U-type [100-1000]) and the gridpoints are ten times more, due to larger integer coefficients. The models solved for AUGMECON2 in Table 6 are driven by |PF*|, being a desirable feature of the algorithm, while the models solved for the original

AUGMECON are driven by the gridpoints' number, which is very high, due to the large coefficients present. The example of Figure 2 is characteristic. It depicts the computational time and the "models solved" parameters for the four 2kp problems with large coefficients (U-type [100-1000]).

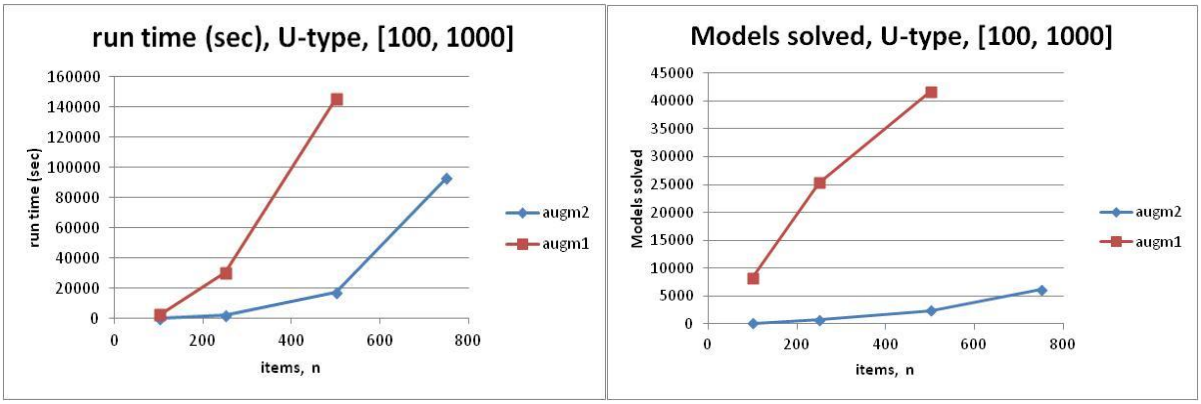

Figure 2. Computational time and "models solved" for the 2kp problems of U-type with objective function coefficients in [100-1000]

This is a key finding of this study, which justifies the improved version of AUGMECON2. Also, the differences are overwhelming in the CPU time in the second part of Table 6 between the two versions. AUGMECON2 is found to be an order of magnitude faster than the original AUGMECON, and also AUGMECON2 solves the hardest 2kp750b dataset, whilst the original AUGMECON cannot solve it in 2 days limit.

By looking at W type [10-100] results (third part of Table 6) we evaluate the newly introduced W type instances. Firstly, the size of the Pareto front in these instances seems to be less than the size of the Pareto front in corresponding U type instances, with the same number of digits for C elements. Also, while for smaller number of items, W type instances seem to be harder than U type instances, the 4 thread use of CPLEX makes even the larger 500 and 750 item datasets affordable. Comparing AUGMECON2 to the original AUGMECON, we see that in these datasets, AUGMECON2 is marginally faster again. The W type [100-1000] instances (fourth part of Table 6) are probably the hardest to solve since both complicating factors are present, W type and 3 digits for C elements. Again, we note that the number of Models solved for AUGMECON2 is driven by the |PF*| number, whilst for original AUGMECON the models solved are driven by the gridpoints which eventually result in a prohibitive number, due to the large coefficients present. AUGMECON2 again solves a dataset, namely 2kp-W-750b that the original AUGMECON cannot solve in 2 days limit.

5.2 Three objective MKP results

Table 7 presents the results for the exact solution of U type instances with 2 digits for C in three objectives and three constraints. From Section 5.1, one concludes that AUGMECON2 dominates AUGMECON in all benchmarks with two objectives and two constraints. This is by far confirmed in the three objective cases. Additionally, in this section we compare AUGMECON2 with method ADECON of Laumanns et al. [5, 39] and MCBB [9, 53]. For large instances ADECON is clearly better than MCBB. Nevertheless, for small to medium instances (up to 30-40 binary variables), MCBB remains the fastest method, but it failed to provide a solution for the 100 binary variables (stopped after 48h). It should be noticed that there is a dominated objective vector among the 6501 Laumanns et al. report on their website. Our results conform to those of Lust at his web site. We find exactly 6500 non dominated vectors for 3kp100. The nadir vectors of these problems which are necessary for the implementation of AUGMECON2, were given as the exact Pareto front of these benchmarks is

available in the literature. As it was mentioned, the runs of AUGMECON2 and AUGMECON were made by the authors in an i5 2.4GHz and the results of the previous methods (computational times of ADECON and MCBB) are adjusted according to the benchmarks of Dongarra [54] and the use of LINPACK utility for standardization of system performance. It must be emphasized that the computational times, although standardized to a common base, they must be used with much caution for inter-method comparisons. On the contrary, they are very useful for examining the evolution of computational time with the size of the problem for each method.

Table 7. AUGMECON2 statistics and comparison with AUGMECON, Adecon, and MCBB for 3-objective MOMKPs (U-Type)

U Type [10,100]	CPU time	Grid points per obj. function	Models solved	|PF*|
AUGMECON2				
3kp40	37 min	540	7802	389
3kp50	159 min	846	24903	1048
3kp100	33 h	1236	103049	6500
AUGMECON				
3kp40	15 h	540	242386	389
3kp50	41 h	846	489746	1048
3kp100	dnt*	1236	dnt	dnt
ADECON [5, 39]				
3kp40	29 min	-	26846	389
3kp50	209 min	-	128695	1048
3kp100	120 h	-	644689	6501
MCBB [9, 53]				
3kp40	6.5 min	-	-	389
3kp50	164 min	-	-	1048
3kp100	dnt	-	-	dnt

*dnt = did not terminate within 48 hours which means problem too big for the method

5.3 Two objective SPP results

Regarding the results for the Bi Objective Set Packing Problem (BOSPP), there are 20 problems with 6 instances each (A, B, C, D, E, F). In their work, Delorme et al. [52] present the mean running time results for the 120 datasets which are presented in Table 8 along with the runs from AUGMECON2. It must be noted that Delorme et al. [52] CPU times are standardized to our system (where AUGMECON2 runs) using Dongarra's benchmarks from LINPACK [54]. We observe that the running times are slightly higher for AUGMECON2 in the case of 100 variables but for the 200 variables they are significantly lower. This finding simply verifies the fact that our approach is practical for BOSPP too, although it cannot provide a rigorous conclusion on whether it is in general faster than Delorme et al. [52] exact procedure. All results for the Bi-Objective Set Packing Problem (BOSPP) are presented analytically (i.e. for every benchmark No 1-20) in an online Appendix in http://sites.google.com/site/kflorios/augmecon2 (in the BOSPP section).

Table 8. Comparison of AUGMECON2 with Delorme et al. [52] mean performance for BOSPP benchmarks No 1-20 averaged over types A-F.

CPU time in seconds							
AUGMECON2							
	A	B	C	D	E	F	Average
100	*21*	*27*	*24*	*17*	*7*	*7*	*17*
200	*2979*	*2596*	*2975*	*2269*	*3591*	*3161*	*2928*
Delorme et al. [52]							
	A	B	C	D	E	F	Average
100	*19*	*24*	*21*	*13*	*7*	*6*	*15*
200	*12256*	*10052*	*10473*	*11328*	*9203*	*12537*	*10975*

5.4 Approximate solution with AUGMECON2 and comparison with SPEA2

Apart from the computation of the exact Pareto set in MOIP problems, we test AUGMECON2 as a provider of approximations of the Pareto set. Approximations of the Pareto set can be obtained much faster than the exact Pareto sets (especially when the required degree of accuracy is low) and they are very common in practice. This is exactly the reason why MOMH algorithms are so popular the last years. An additional advantage of AUGMECON2 is that it can be used also as an approximate method with an adjustable degree of approximation. The tunable parameter is the resolution of the grid for every constrained objective. Defining the number of intervals g_k (which means g_k+1 grid points) for k=2,3,…,p we can obtain a representation of the Pareto set with a predetermined density. We will examine the performance of AUGMECON2 in computing approximations of the Pareto set in benchmark problems that we used before and we will compare it with SPEA2 [49], a very popular MOMH method. The SPEA2 algorithm implemented in PISA [55, 56] has been employed as a standard MOMH for this problem. Our aim is to show that the use of AUGMECON2 which is a general purpose algorithm is competitive with a general purpose MOMH algorithm like SPEA2. It must be noted that in a previous work of ours, SPEA2 was found superior to NSGAII for MOMKP problems [9].

5.4.1 MOMKP problem with 3 objective functions

We will apply AUGMECON2 for providing an approximation of the Pareto set in the 3kp100 problem elaborated in section 5.2. In Table 9 we show the evolution of the Pareto set representation as a function of grid points that we divide the ranges of the second and third objective function. We also record the computational time in a system core i5, 2.40 GHz and 4GB RAM running Windows 7 32-bit. The efficiency of the representation is expressed by the coverage metric C(A,B). The coverage metric, in our case, presents the percentage of Pareto optimal solutions in set B, which are weakly dominated by a solution discovered by the approximate algorithm in set A. The term "weakly" is used to incorporate the cases of identical solutions in the two sets ([8]; p. 325).

$$C(A,B) = \frac{\left|\left\{b \in B \mid \exists a \in A : a \geq^{w} b\right\}\right|}{|B|} \tag{5}$$

the symbol $\geq^{w}$ represents weak dominance, that also holds true if f(a) = f(b). Therefore, the coverage metric C(AUGMECON2, EPS) indicates how many solutions from AUGMECON2 are also found in the Exact Pareto Set (EPS). It actually reports the % of discovered Pareto optimal solutions by

AUGMECON2 and the closer to 1 is the coverage metric C(AUGMECON2, EPS) the better is the representation. The number of gridpoints varies from 10 to 700. From Table 9 we can see that with 400 gridpoints per objective function we can achieve a rough 80% coverage in 35,446 seconds. Going to 90% coverage means almost doubling the computational time (almost 63,708 seconds).

Table 9. AUGMECON2 statistics for 3kp100 dataset of Laumanns et al. [39].

No. Grid points	C(AUGMECON2, EPS)	CPU time (sec)
10	0.0103	53
20	0.0348	201
30	0.0685	383
40	0.1094	678
50	0.1489	979
60	0.1954	1389
70	0.2329	1836
80	0.2815	2400
90	0.3109	2752
100	0.3546	3691
120	0.4284	5094
140	0.4736	7140
160	0.5213	8458
180	0.5659	10465
200	0.5998	12010
240	0.6670	16334
280	0.7162	19198
320	0.7410	23345
360	0.7860	27234
400	0.8091	35446
460	0.8371	42281
520	0.8765	48043
580	0.8949	56498
640	0.9048	63708
700	0.9126	70429

Subsequently we applied the popular SPEA2 algorithm for calculating a representation of the Pareto set in the specific problem using the PISA platform ([55, 56]) and the GNU gcc 4.6.2 C compiler. The SPEA2 algorithm with a population of 7000 chromosomes was executed for 1000 generations, taking 42840 seconds (=11h 54min) to run. The SPEA2 algorithm produced 1719 distinct solutions in the final population of 7000 chromosomes. Out of these, 1390 were dominated by the exact Pareto front of 6500 solutions and 329 were the actually nondominated objective vectors. Thus, the Coverage metric is computed as C(SPEA2, EPS)=329/6500=5.06%. If we look at Table 9, we see that the heuristic application of AUGMECON2 with sparse grids is capable of producing an approximation to the exact Pareto front with Coverage(AUGMECON2,EPS)= 0.0685 in CPU time=383sec (using only 30 gridpoints per objective function). Even if we want to consider all 1719 solutions of SPEA2 as practically Pareto optimal which yields a value of Coverage=1719/6500=26.45%, we see from Table 9, that AUGMECON2 would produce such an approximation in only 2400sec with a Coverage=0.2815 (using only 80 grid points per axis). So we see that in 40minutes, a heuristic use of AUGMECON2 yields a higher quality result than 11h 54min of computation with SPEA2 in the same hardware.

5.4.2 MOMKP problem with 2 objective functions

We also applied AUGMECON2 and SPEA2 in the knapsack problems with two objective functions that we used in section 5.1 and specifically in the U instances with 2 digits (2kp100, 2kp250, 2kp500, 2kp750). We fix the number of gridpoints to 800 for all four runs as we aimed at approximations of the Pareto set (the number of gridpoints for the exact Pareto set is available in Table 6 for the four cases). The calculation time and the coverage for the four cases are depicted in Table 10:

Table 10. Approximations of the Pareto set with AUGMECON2 (800 gridpoints)

Problem	C(AUGMECON2, EPS)	CPU time (sec)
2kp100	0.9835	39
2kp250	0.6884	630
2kp500	0.4025	3324
2kp750	0.2092	12195

Subsequently we solved the four problems with SPEA2 using the PISA platform. After some preliminary experiments we used the parameters shown in Table 11 that provide best results:

Table 11. Parameters used in SPEA2 simulations (following PISA terminology).

Parameter	2kp100	2kp250	2kp500	2kp750	3kp100
Alpha	1500	2000	2500	3000	7000
Mu	750	1000	1250	1500	3500
Lambda	750	1000	1250	1500	3500
Maxgen	1000	3000	3000	2000	1000
Dim	2	2	2	2	3
Cost evaluations	3,000,000	12,000,000	15,000,000	12,000,000	21,000,000

In the above table Alpha is the population size, Mu is the number of parents, Lambda is the number of children, Maxgen is the maximum number of generations, Dim is the number of objective functions and constraints and Cost evaluations = Dim × Maxgen × Alpha. Using the above parameters the calculation time and the coverage for the four cases are depicted in Table 12:

Table 12. Approximations of the Pareto set with SPEA2

Problem	C(SPEA2, EPS)	CPU time (sec)
2kp100	0.5950	1799
2kp250	0.0000	10258
2kp500	0.0000	15780
2kp750	0.0000	15179

We can observe that except the smallest problem the coverage achieved by SPEA2 is zero. This means that all the solutions produced are dominated by the Pareto optimal solutions of the exact Pareto set. The outperformance of AUGMECON2 over SPEA2 is evident and it can be better illustrated also in terms of the produced Pareto fronts as shown in Figure 3.

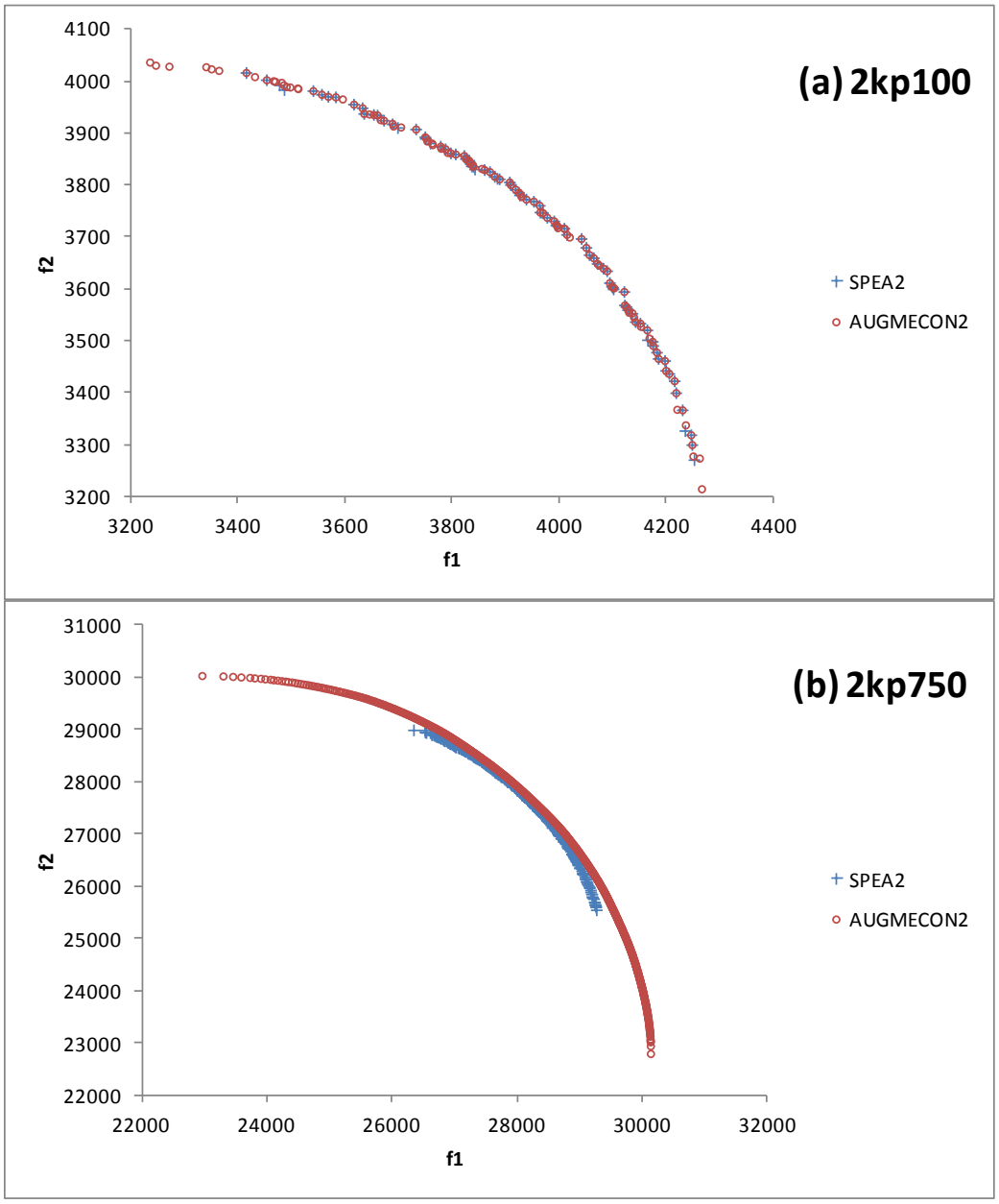

Figure 3. *Visualization of Pareto front approximations for SPEA2 and AUGMECON2 for the two extreme cases (2kp100 and 2kp750)*

It is observed from Figure 3 that the SPEA2 implementation for MOMKP suffers from the 'middling effect' (fail to catch the Pareto optimal solutions at the two edges), which is caused by the greedy repair technique in the knapsack module.

6. Case study: R&D project selection in a University

The selection of R&D projects is a complex problem, which is faced by universities, research institutes, firms etc on a regular basis. In this study, this problem is modeled as a multi-objective optimization problem, and particularly a Multi-objective Integer Programming problem (MOIP). The a posteriori approach to multi-objective optimization is adopted aiming at generating exactly all Pareto optimal solutions of the problem. Mavrotas et al. [57] solved this problem with a combined MCDA and IP approach. In the current work we follow a different approach. We avoid expressing criterion weights information and solve the problem in a truly multi-objective fashion. The aim is to provide the

decision makers as much information as possible before they express their preferences (weights, etc). The statement of the problem is as follows:

A Greek university has to decide on funds distribution among 150 candidate R&D projects. Every project belongs in one of the nine departments of the university. Furthermore, every project is classified as basic or applied research. The available budget is 6,000,000 Euros. Every project has obviously a relevant cost associated with it. The evaluation phase by a committee assigned scores in the range 1 to 10 for every project for three criteria: Innovation, Usefulness and Faculty Sufficiency. The data of the problem are available in [57].

We model the problem as a MOIP. The objective functions are total Innovation, Usefulness and Faculty Sufficiency scores associated with a solution or decision by the university. Decision variables are 0-1 variables equal to 1 if a project is approved and 0 if not. There are three objective functions to take into account, so this is a multi-objective optimization problem. No weight information is asked for by the DM at this point of analysis. Also, no equal weights are assumed. There is a budget constraint of 6,000,000€ available funds. Also, there are important policy (or segmentation) constraints dictated by the R&D policy of the university. Funds approved to every department must be in a certain range for every department, as a percentage of the total approved budget of the Programme. This ensures that all departments, more or less, obtain some funds. Also, that no super department accumulates all funds instead of another weaker department. These constraints are stated as lower and upper bounds on the total approved budget of the Programme. These bounds are different for every department and are proportional to e.g. the size of the department. Also, there is a statement that the basic research must be protected. So at least, 30% of the approved projects must come from the basic research type. While lower and upper bounds for departments are based on a percentage of total cost, this lower bound on basic research is based on a percentage of the number of approved projects.

The MOIP model is as follows:

$$Max \sum_{i=1}^{N} in_i x_i$$

$$Max \sum_{i=1}^{N} use_i x_i$$

$$Max \sum_{i=1}^{N} fs_i x_i$$

st

$$\sum_{i=1}^{N} \cos t_i x_i \le budg \qquad (6)$$

$$\sum_{i \in d(k)} \cos t_i x_i \ge lb_k \cdot \sum_{i=1}^{N} \cos t_i x_i, \quad k = 1,...,N_D$$

$$\sum_{i \in d(k)} \cos t_i x_i \le ub_k \cdot \sum_{i=1}^{N} \cos t_i x_i, \quad k = 1,...,N_D$$

$$\sum_{i \in basic} x_i \ge lb_{basic} \cdot \sum_{i=1}^{N} x_i$$

The variables are $x_i \in \{0,1\}$ and the various parameters are in_i, use_i, fs_i the Innovation, Usefulness and Faculty Sufficiency scores of every project in a range 1-10 (data), $cost_i$ the cost associated with every

project in €, *d(k)* is the set of projects from *k*-th department (*k* in {1,2,3,..,N_D}), *basic* is the set of basic research projects, lb_k is a decimal in [0, 1] as a lower bound for funds approved to department k as a portion of total funds, ub_k a decimal in [0, 1] as an upper bound for funds approved to department k as a portion of total funds, lb_{basic} a decimal in [0, 1] as a lower bound for the number of projects approved to basic research as a portion of the total number of projects approved. In the following lb_{basic}=0.30, N_D=9, *N*=150, *budg*=6,000 k€ and lb_k and ub_k, *k*=1,…, N_D are as in [57].

The model MOIP is solved with AUGMECON2 in GAMS 23.5 using CPLEX 12.2 in order to produce the exact Pareto set. This means that all the Pareto optimal project portfolios based on these three objective functions are generated, or, in other words, there is no Pareto optimal project portfolio that is left undiscovered. In total, 4075 Pareto optimal solutions (project portfolios) are generated. The computational time is 7,633 seconds and 11,477 IP models are solved. It must be noted that using the original AUGMECON method the computational time was 19,522 seconds and 29,122 IP models were solved. The coefficients for the three objective functions are integer and the calculated objective function ranges for the second and third objective function is 180 for both of them. For every one of the 4075 Pareto optimal solutions we calculate some useful indicators such as the share of budget for each department (*ShareBudgetDept(k)*), the shares of Applied and Basic research projects, the total number of projects approved and the consumed budget. In Table 13 we summarize the statistics of these indicators.

Table 13. Indicators statistics for the Pareto optimal solutions of MOIP (|PF*|=4075)

	mean	std	min	max	Median
ShareBudgetDept(%)					
A	14.22	0.31	14	15.77	14.14
B	16.72	1.21	12.34	18.98	16.51
C	7.05	0.84	3.87	8.74	7.35
D	12.62	0.87	9.48	14	12.86
E	6.79	1.12	4.3	8.81	6.22
F	11.69	1.37	10.02	17.94	11
G	9.77	0.37	9	10.7	10.03
H	8.57	1.30	4.77	10.18	8.44
I	12.56	1.24	9	15.42	12.39
Ratio of projects(%)					
Applied	54.39	1.76	50	61.25	54.22
Basic	45.61	1.76	38.75	50	45.78
Budget Approved	5995	9.18	5940	6000	5998
Projects Approved	82.08	1.49	77	86	82

In a decision making process the calculation of the exact Pareto front is the first step towards the final choice. The decision makers have a set of candidate solutions (project portfolios) among which they must select. No other project portfolio except those in the Pareto front may be considered by a rational decision maker. In other words, it may be used to deter inferior solutions (that may be suggested exogenously) from being adopted. Therefore the generation of the exact Pareto front by AUGMECON2 may be considered as a first screening of options that provide fruitful information to the decision maker. For example, we can observe which projects are present in all Pareto optimal portfolios and which are in none of them. We call the former '*elite'* projects (projects that are anyway

selected, independently of which Pareto optimal solution will eventually be selected) and the latter '*junk'* projects (projects that will be never selected, independently of which Pareto optimal solution will eventually be selected). In the specific problem we found out that we have 27 'elite' projects and 23 'junk'. If we relax the relevant conditions (e.g. 'elite' projects should be considered those that appear in more than 99% or 95% of the Pareto optimal solutions and correspondingly 'junk' those appear in less than 1% or 5%) these sets can be even more populated. If we sort the projects by their participation frequency in the Pareto Front (PF*) and then create the relative graph we obtain useful information as depicted in Figure 4.

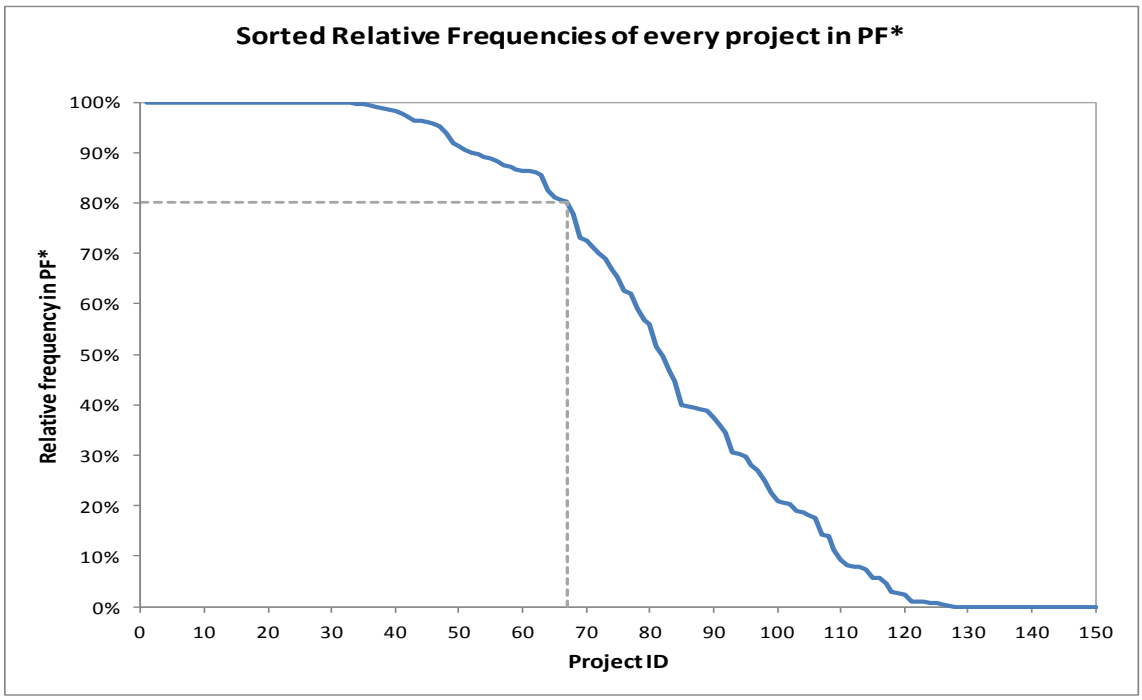

Figure 4. *Relative frequencies of every project in the Pareto optimal population of solutions (|PF*|=4075).*

By looking at Figure 4, we can see the 27 'elite' projects that have 100% relative frequency (top horizontal line) and the 23 projects with zero frequency ('junk'). By drawing a vertical line we can see the frequency of each project (it is reminded that they are sorted according to their relative frequency on the Pareto optimal portfolios). In addition, by drawing a horizontal line to e.g. 80% we can observe that 67 projects appear in the 80% of the Pareto optimal portfolios. Subsequently, the decision makers, exploiting this information, they can proceed using an interactive approach (like e.g. the interactive filtering introduced by Steuer in [1]) to their final selection. However the description of this process is beyond the scope of the present paper.

7. Concluding remarks

The aim of this paper is to propose and evaluate an enhancement of the original augmented ε-constraint method which is especially suitable to cope with MOIP problems. Specifically, the new version AUGMECON2 proved to be very efficient in providing the exact Pareto set in MOIP problems compared to the previous version and some other methods in the literature. The basic innovation in AUGMECON2 in relation to the original version is the introduction of the bypass

coefficient. The bypass coefficient exploits the slack/surplus information of the innermost constrained objective function and skips an often considerable number of grid points before it proceeds to the next call to the solver. This is done with no cost whatsoever to the accuracy of the algorithm as only redundant iterations resulting in the same Pareto optimal solution are skipped. So, being a win-win feature, the bypass coefficient is proven a valuable characteristic of AUGMECON2 over AUGMECON as shown in the computational experiment with several MOIP problems. In the computational experiments we used known benchmarks found in the literature as well as some new benchmarks that are published for first time. The problems in which the performance of the proposed method is evaluated is a well studied MOIP problem, namely the Multi-Objective Multidimensional Knapsack Problem (MOMKP) and also the Bi-Objective Set Packing Problem (BOSPP). Finally the AUGMECON2 method is also tested in a real case study from the literature where it provides the exact Pareto set in a project selection problem regarding research grants.

Regarding the MOMKP, the proposed method solved successfully difficult bi-objective instances of the MOMKP, both uncorrelated and weakly correlated. It greatly improves the original version AUGMECON and this is more apparent where larger integer coefficients for the objective functions are present. The key finding is that the "models solved" metric for AUGMECON2 is linear in the size of the Pareto front in all bi-objective MOMKP instances, while for the original AUGMECON method the "models solved" metric is proportional to the number of grid points which is actually the range of the payoff table for the constrained objective. Especially in the case of 3 digit objective function coefficients (i.e. in the range [100, 1000]) the economies of AUGMECON2 over AUGMECON are impressive. As it is obvious, the 3 digit objective function coefficient problems are harder to solve than the regular 2 digit coefficients problems. In addition, it is also noticed that the newly introduced W type instances are harder to solve than the regular U type instances, although the cardinality of their Pareto front is noticeably smaller. The three objective problems are much more challenging than bi-objective problems. For three objectives and three constraints, only up to 100 items are solved, in high computational time. We tested AUGMECON2 in benchmark problems found in the literature and we compared it also with two other exact methods, namely, the adaptive ε-constraint (ADECON) and MCBB. The results were favourable for AUGMECON2 especially in the larger instances.

Regarding the BOSPP, the AUGMECON2 method is compared with the dichotomic procedure in [41]. The results are encouraging, even after taking into account the faster hardware used in our experiments through the use of the Linpack benchmark [54]. The performance of AUGMECON2, especially in the larger datasets, is a positive sign for its applicability in demanding MOIP benchmarks, as the BOSPP instances of [41], available also at the MOCOlib of X.Gandibleux.

In addition to the use of AUGMECON2 as an exact method, a heuristic use of the proposed method is made in order to compute approximations to the Pareto front. Our method proves very practical, compared to an available implementation of SPEA2 within the PISA framework ([55, 56]). Nevertheless, comparison to specialized MOMH for MOMKP was not undertaken, because we wanted to consider only widely available general purpose MOMH techniques, such as SPEA2. Used as a heuristic technique using sparse grids, AUGMECON2 is faster and more accurate than SPEA2/PISA in MOMKP.

Finally, in order to test AUGMECON2 in practical MOIP problems with no special structure, a case study concerning a R&D Project Selection problem at a University i.e. a typical capital budgeting problem is solved exactly with the proposed method. This is a MOIP problem with different structure than those studied before (MOMKP, BOSPP). It has three objective functions, 150 binary variables and complex policy, logical and segmentation constraints. The model is coded and solved in GAMS.

The exact Pareto front is generated in almost two hours containing 4075 Pareto optimal project portfolios. This means that AUGMECON2 can be effectively used in large practical MOIP problems with no specific structure providing the exact Pareto set.

Future research includes methodological and technical issues. The methodological issues have to do with the further exploitation of the information provided in every iteration of the ε-constraint method (to create bypass rules not only for the innermost loop) and the safe calculation of lower bounds for the nadir values. The technical issues have to do with coding AUGMECON2 in other than GAMS environments (MATLAB, C) and directly connect it with solvers' dlls like CPLEX and GUROBI. Parallelization of the process is also within our future plans as well as testing of the method in other MOIP problems like e.g. Multi-Objective Set Covering problems, Multi-Objective Shortest Path problems and Multi-Objective Spanning Tree problems.

References

[1] R.E. Steuer, Multiple Criteria Optimization.Theory, Computation and Application, Krieger, Malabar FL, 1986.

[2] C.L. Hwang, A. Masud, Multiple Objective Decision Making. Methods and Applications: A state of the art survey, Lecture Notes in Economics and Mathematical Systems Vol. 164, Springer-Verlag, Berlin, 1979.

[3] G. Mavrotas, Effective implementation of the ε-constraint method in multi-objective mathematical programming problems, Applied Mathematics and Computation 213 (2009) 455-465.

[4] M. Ehrgott, D.M. Ryan, Constructing Robust Crew Schedules with Bicriteria Optimization, Journal of Multi-Criteria Decision Analysis 11 (2002) 139-150.

[5] M. Laumanns, L. Thiele, E. Zitzler, An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method, European Journal of Operational Research 169 (2006) 932-942.

[6] H.W. Hamacher, C.R. Pedersen, S. Ruzika, Finding representative systems for discrete bicriterion optimization problems, Operations Research Letters 35 (2007) 336-344.

[7] C.A. Coello Coello, D.A. Van Veldhuizen, G.A. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, Boston, MA, 2002.

[8] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley and Sons, Chichester, 2001.

[9] K. Florios, G. Mavrotas, D. Diakoulaki, Solving multi-objective, multi-constraint knapsack problems using mathematical programming and evolutionary algorithms, European Journal of Operational Research 203 (2010) 14-21.

[10] A. Chinchuluun, P.M. Pardalos, A survey of recent developments in multiobjective optimization, Annals of Operations Research 154 (2007) 29-50.

[11] G.R. Bitran, Linear multiple objective programs with zero-one variables, Mathematical Programming 13 (1977) 121-139.

[12] G.R. Bitran, Theory and algorithms for linear multiple objective programs with zero-one variables, Mathematical Programming 17 (1979) 362-390.

[13] D. Klein, E. Hannan, An algorithm for the multiple objective integer linear programming problem, European Journal of Operational Research 9 (1982) 378-385.

[14] L.G. Chalmet, L. Lemonidis, D.J. Elzinga, An algorithm for the bi-criterion integer programming problem, European Journal of Operational Research 25 (1986) 292-300.

[15] G.R. Jahanshahloo, F. Hosseinzadeh, N. Shoja, G. Tohidi, A method for generating all efficient solutions of 0-1 multi-objective linear programming problem, Applied Mathematics and Computation 169 (2005) 874-886.

[16] F. Sourd, O. Spanjaard, A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem, INFORMS Journal on Computing 20(3) (2008) 472-484.

[17] T.K. Ralphs, M.J. Saltzman, M.M. Wiecek, An improved algorithm for solving biobjective integer programs, Annals of Operations Research 147 (2006) 43-70.

[18] M. Ozlen, M. Azizoglu, Multi-objective integer programming: A general approach for generating all non-dominated solutions, European Journal of Operational Research 199 (2009) 25-35.

[19] A. Przybylski, X. Gandibleux, M. Ehrgott, A two-phase method for multi-objective integer programming and its application to the assignment problem with three objectives, Discrete Optimization 7 (2010) 149-165.

[20] Y.Y. Haimes, L.S. Lasdon, D.A. Wismer, On a bicriterion formulation of the problems of integrated system identification and system optimization, IEEE Transactions on Systems, Man, and Cybernetics 1 (1971) 296-297.
[21] D. Rayside, H.C. Estler, D. Jackson, The guided improvement algorithm for exact, general-purpose, many-objective combinatorial optimization, Massachusetts Institute of Technology Technical Report, Computer Science and Artificial Intelligence Laboratory, 2009.
[22] M. Ehrgott, S. Ruzika, Improved ε-constraint method for multiobjective programming, Journal of Optimization Theory and Applications 138 (2008) 375-396.
[23] J. Sylva, A. Crema, A method for finding the set of non-dominated vectors for multiple objective integer linear programs, European Journal of Operational Research 158 (2004) 46-55.
[24] J. Sylva, A. Crema, A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs, European Journal of Operational Research 180 (2007) 1011-1027.
[25] B. Lokman, Approaches for multi-objective combinatorial optimization problems, MSc Thesis, Middle East Technical University, Ankara, 2007.
[26] M. Koksalan, B. Lokman, Approximating the nondominated frontiers of multi-objective combinatorial optimization problems, Naval Research Logistics 56 (2009) 191-198.
[27] J. Teghem, P.L. Kunsch, A survey of techniques for finding efficient solutions to multi-objective integer linear programming, Asia-Pacific Journal of Operational Research 3 (1986) 95-108.
[28] E.L. Ulungu, J. Teghem, Multi-objective combinatorial optimization problems: A survey, Journal of Multi-Criteria Decision Analysis 3 (1994) 83-104.
[29] L.M. Rasmussen, Zero-one programming with multiple criteria, European Journal of Operational Research 26 (1986) 83-95.
[30] M. Ehrgott, X. Gandibleux, A survey and annotated bibliography of multiobjective combinatorial optimization, OR Spectrum 22 (2000) 425-460.
[31] M.J. Alves, J. Climaco, A review of interactive methods for multiobjective integer and mixed-integer programming, European Journal of Operational Research 180 (2007) 99-115.
[32] A. Jaszkiewicz, On the computational efficiency of multiple objective metaheuristics. The knapsack problem case study, European Journal of Operational Research 158 (2004) 418-433.
[33] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, IEEE Transactions on Evolutionary Computation 3 (1999) 257-271.
[34] K. Klamroth, M.M. Wiecek, Dynamic programming approaches to the multiple criteria knapsack problem, Naval Research Logistics 47 (2000) 57-76.
[35] T. Erlebach, H. Kellerer, U. Pferschy, Approximating multiobjective knapsack problems, Management Science 48 (2002) 1603-1612.
[36] R. Shah, P. Reed, Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems, European Journal of Operational Research 211 (2011) 466-479.
[37] T. Lust, J. Teghem, The multiobjective multidimensional knapsack problem: a survey and a new approach, International Transactions in Operational Research 19 (2012) 495-520.
[38] A. Jaszkiewicz, On the performance of multiple-objective genetic local search on the 0/1 knapsack problem – A comparative experiment, IEEE Transactions on Evolutionary Computation 6 (2002) 402-412.
[39] M. Laumanns, L. Thiele, E. Zitzler, 2005. An adaptive scheme to generate the Pareto front based on the epsilon-constraint method, in: J. Branke, K. Deb, K. Miettinen, R. Steuer (Eds.), Practical Approaches to Multi-Objective Optimization, Dagstuhl Seminar Proceedings, Vol. 04461, URN: urn:nbn:de:0030-drops-2465, URL: http://drops.dagstuhl.de/opus/volltexte/2005/246.
[40] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, 1999.
[41] X. Delorme, X. Gandibleux, F. Degoutin, Evolutionary, constructive and hybrid procedures for the bi-objective set packing problem, European Journal of Operational Research 204 (2010) 206-217.
[42] F. Tricoire, Multi-directional local search, Computers & Operations Research 39 (2012) 3089-3101.
[43] H. Isermann, R.E. Steuer, Computational experience concerning payoff tables and minimum criterion values over the efficient set, European Journal of Operational Research 33 (1987) 91-97.
[44] M.I. Dessouky, M. Ghiassi, W.J. Davis, Estimates of the mínimum nondominated criterion values in multiple criteria decisión making, Engineering Costs and Production Economics 10 (1986) 95-104.
[45] B. Metev, V. Vassilev, A method for nadir point estimation in MOLP problems, Cybernetics and Information Technologies 3 (2003) 15-24.
[46] M.J. Alves, J.P. Costa, An exact method for computing the nadir values in multiple objective linear programming, European Journal of Operational Research 198 (2009) 637-646.
[47] J.M. Jorge, An algorithm for optimizing a linear function over an integer efficient set, European Journal of Operational Research 195 (2009) 98-103.

[48] A. Brooke, D. Kendrick, A. Meeraus, R. Raman, GAMS. A user's guide, GAMS development corporation, Washington, 1998.
[49] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, in: K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, T. Fogarty (Eds.), Evolutionary Methods for Design, Optimization, and Control, Barcelona, CIMNE , Spain, 2002, pp. 19-26.
[50] C.H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity, Dover, New York, 1998.
[51] F. Degoutin, Problemes bi-objectifs en optimization combinatoire et capacite d'infrastructures ferroviaires. Master Thesis. Universite de Valenciennes et du Hainaut Cambresis, Valenciennes, France, 2002 (in French).
[52] X. Delorme, X. Gandibleux, F. Degoutin, Résolution approchée du problème de set packing bi-objectifs. In Proceedings de l'École d'Automne de Recherche Opérationnelle de Tours (EARO), October 2003, pp. 74-80 (in French).
[53] G. Mavrotas, D. Diakoulaki, Multi-criteria branch & bound: A vector maximization algorithm for Mixed 0-1 Multiple Objective Linear Programming, Applied Mathematics and Computation 171 (2005) 53-71.
[54] J.J. Dongarra, Performance of Various Computers Using Standard Linear Equations Software, Linpack Benchmark Report, University of Tennessee Computer Science Technical Report, CS-89-85, 2008.
[55] S. Bleuler, M. Laumanns, L. Thiele, E. Zitzler, PISA – A Platform and Programming Language Independent Interface for Search Algorithms, in: C. Fonseca, P. Fleming, E. Zitzler, K. Deb, L. Thiele (Eds.), Evolutionary Multi-Criterion Optimization, Springer, Heidelberg, 2003, pp. 494-508.
[56] S. Bleuler, M. Laumanns, L. Thiele, E. Zitzler 2003. The PISA Homepage. Available online at http://www.tik.ee.ethz.ch/pisa.
[57] G. Mavrotas, D. Diakoulaki, A. Kourentzis, Selection among ranked projects under segmentation, policy and logical constraints, European Journal of Operational Research 187 (2008) 177-192.